About the Final

- Open book and open notes
  - But you won’t have time to read during final!
  - No laptops, no mobile devices
- Topics:
  - Lectures 16 through the end of quarter
  - Projects 3 and 4
  - HW3
- Sample finals on course website!

XML Outline

- What is XML?
- Syntax
- Semistructured data
- DTDs
- XPath

What is XML?

- Stands for eXtensible Markup Language
- Applications:
  - Data exchange
  - Un-normalized data
- Other usages:
  - Configuration files: e.g. Web.Config
  - Document markup: e.g. XHTML
- Roots: SGML - a very nasty language
  - We will study only XML as data

Readings

New edition: Sections 11.1 – 11.3 and 12.1
Old edition: Subset of material in 4.6 and 4.7
(coverage of XML is better in new version)
Data Exchange

- Relational data does not have a syntax
  - I can’t “give” you my relational database or parts of it
  - Need some file format: CSV (comma-separated-values), ASN.1
- XML
  - Is a more advanced file format
  - Also has its own data model: semistructured
- Main idea: applications exchange information in XML

From HTML to XML

HTML describes the presentation

HTML

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>  
Abiteboul, Hull, Vianu  
<br> Addison Wesley, 1995  
</p>  
<p> <i> Data on the Web </i>  
Abiteboul, Buneman, Suciu  
<br> Morgan Kaufmann, 1999  
</p>
```

XML Syntax

```
<bibliography>
    <book>    <title> Foundations… </title>  
    <author> Abiteboul </author>  
    <author> Hull </author>  
    <author> Vianu </author>  
    <publisher> Addison Wesley </publisher>  
    <year> 1995 </year>  
</book>  
</bibliography>
```

XML describes the content

XML Terminology

- Tags: book, title, author, ...
- Elements are nested
- Empty element: <red></red> abbrv. <red/>
- An XML document: single root element

Well-Formed XML

```
<? xml version="1.0" encoding="utf-8" standalone="yes" ?>  
<SomeTag>  
    ...  
</SomeTag>
```
More XML: Attributes

```xml
<book price="55" currency="USD">
  <title>Foundations of Databases</title>
  <author>Abiteboul</author>
  ...
  <year>1995</year>
</book>
```

Attributes v.s. Elements

```xml
<book>
  <title>Foundations of Databases</title>
  <author>Abiteboul</author>
  ...
  <year>1995</year>
  <price>55</price>
  <currency>USD</currency>
</book>
```

Attributes are alternative ways to represent data.

Comparison

<table>
<thead>
<tr>
<th>Elements</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered</td>
<td>Unordered</td>
</tr>
<tr>
<td>May be repeated</td>
<td>Must be unique</td>
</tr>
<tr>
<td>May be nested</td>
<td>Must be atomic</td>
</tr>
</tbody>
</table>

XML v.s. HTML

- What are the differences between XML and HTML?
  - HTML may be non-well formed: e.g. `<br>` without `</br>`. Better: `<br/>`; XML must be well formed.
  - HTML has semantics: `<br>` means newline, `<i>` means italic etc. XML has no semantics.

XML Semantics: a Tree!

```xml
<data>
  <person id="o555">
    <name>Mary</name>
    <address>
      <street>Maple</street>
      <no>345</no>
      <city>Seattle</city>
    </address>
  </person>
  <person>
    <name>John</name>
    <address>Thailand</address>
    <phone>23456</phone>
  </person>
</data>
```

Order matters!!!

XML Data

- XML is self-describing
- Schema elements become part of the data
  - Relational schema: `person(name, phone)`
  - In XML `<person>`, `<name>`, `<phone>` are part of the data, and are repeated many times.
- Consequence: XML is much more flexible
- XML = semistructured data
Mapping Relational Data to XML Data

The canonical mapping:

```
<people>
  <person>
    <name>John</name>
    <phone>3634</phone>
  </person>
  <person>
    <name>Sue</name>
    <phone>6343</phone>
  </person>
  <person>
    <name>Dick</name>
    <phone>6363</phone>
  </person>
</people>
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>3634</td>
</tr>
<tr>
<td>Sue</td>
<td>6343</td>
</tr>
<tr>
<td>Dick</td>
<td>6363</td>
</tr>
</tbody>
</table>
```

Application specific mapping:

```
<people>
  <person>
    <name>John</name>
    <phone>3634</phone>
    <order>
      <date>2002</date>
      <product>Gizmo</product>
    </order>
    <order>
      <date>2004</date>
      <product>Gadget</product>
    </order>
  </person>
  <person>
    <name>Sue</name>
    <phone>6343</phone>
    <order>
      <date>2004</date>
      <product>Gadget</product>
    </order>
  </person>
</people>
```

```
<table>
<thead>
<tr>
<th>PersonName</th>
<th>Date</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>2002</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>2004</td>
<td>Gadget</td>
</tr>
</tbody>
</table>
```

XML is Semi-structured Data

- Missing attributes:
  - `<person>` `<name>John</name>` `<phone>1234</phone>`
  - No phone!

- Could represent in a table with nulls:
  - `Name: John`, `phone: 1234` for `Name: Joe` with no phone.

- Repeated attributes:
  - `<person>` `<name>Mary</name>` `<phone>2345</phone>` `<phone>3456</phone>`
  - Two phones!

- Impossible in tables:
  - `name: Mary`, `phone: 2345`, `phone: 3456`, `???

XML is Semi-structured Data

- Attributes with different types in different objects:
  - `<person>` `<name>` `<first>John</first>` `<last>Smith</last>`
  - `<name>` `<phone>1234</phone>`

- Nested collections (no 1NF):
  - Heterogeneous collections:
    - `<db>` contains both `<books>` and `<publisher>`s

- Structure:
  - `<person>` `<name>` `<phone>`
Document Type Definitions (DTD)

- Part of the original XML specification
- An XML document may have a DTD
- XML document:
  - Well-formed = if tags are correctly closed
  - Valid = if it has a DTD and conforms to it
- Validation is useful in data exchange

Goals:
- Define what tags and attributes are allowed
- Define how they are nested
- Define how they are ordered

Superseded by XML Schema (Book Sec. 11.4)
- Very complex: DTDs still used widely

Very Simple DTD

```xml
<!DOCTYPE company [ 
  <!ELEMENT company   ((person|product)*)>  
  <!ELEMENT person  (ssn, name, office, phone?)>  
  <!ELEMENT ssn        (#PCDATA)>  
  <!ELEMENT name     (#PCDATA)>  
  <!ELEMENT office     (#PCDATA)>  
  <!ELEMENT phone    (#PCDATA)>  
  <!ELEMENT product  (pid, name, description?)>  
  <!ELEMENT pid    (#PCDATA)>  
  <!ELEMENT description    (#PCDATA)> ]>
```

Example of valid XML document:

```xml
<company>
  <person>
    <ssn>123456789</ssn>
    <name>John</name>
    <office>B432</office>
    <phone>1234</phone>
  </person>
  <person>
    <ssn>987654321</ssn>
    <name>Jim</name>
    <office>B123</office>
  </person>
  <product>...</product>
</company>
```

DTD: The Content Model

```xml
<!ELEMENT tag (CONTENT)>  
```

- Content model:
  - Complex = a regular expression over other elements
  - Text-only = #PCDATA
  - Empty = EMPTY
  - Any = ANY
  - Mixed content = (#PCDATA | A | B | C)*

DTD: Regular Expressions

- Sequence
  ```xml
  <ELEMENT name (firstName, lastName)>  
  ```
- Optional
  ```xml
  <ELEMENT name (firstName?, lastName)>  
  ```
- Kleene star
  ```xml
  <ELEMENT person (name, phone*)>  
  ```
- Alternation
  ```xml
  <ELEMENT person (name, phone|email)>  
  ```
Querying XML Data

- XPath = simple navigation through the tree
- XQuery = the SQL of XML
- XSLT = recursive traversal
  - will not discuss in class

Sample Data for Queries

```
<bib>
  <book>
    <publisher> Addison-Wesley </publisher>
    <author> Serge Abiteboul </author>
    <author> Rick Hull </author>
    <title> Foundations of Databases </title>
    <year> 1995 </year>
  </book>
  <book>
    <publisher> Freeman </publisher>
    <author> Jeffrey D. Ullman </author>
    <title> Principles of Database and Knowledge Base Systems </title>
    <year> 1998 </year>
  </book>
</bib>
```

Data Model for XPath

XPath returns a sequence of items. An item is either:
- A value of primitive type, or
- A node (doc, element, or attribute)

```
//author
```

XPath: Simple Expressions

```
/bib/book/year
Result: <year> 1995 </year>
        <year> 1998 </year>

/bib/paper/year
Result: empty          (there were no papers)
```

XPath: Restricted Kleene Closure

```
//author
Result: <author> Serge Abiteboul </author>
        <author> Rick Hull </author>
```

```
/bib/first-name
Result: <first-name> Rick </first-name>
```
XPath: Attribute Nodes

`/bib/book/@price`
Result: "55"

@price means that price has to be an attribute

XPath: Wildcard

```
/author/*
```
Result: <first-name> Rick </first-name>
<last-name> Hull </last-name>

* Matches any element
@* Matches any attribute

XPath: Text Nodes

```
/bib/book/author/text()
```
Result: Serge Abiteboul
Victor Vianu
Jeffrey D. Ullman

Rick Hull doesn’t appear because he has first-name, last-name

Functions in XPath:

- `text()` = matches the text value
- `node()` = matches any node (= * or @* or text())
- `name()` = returns the name of the current tag

XPath: Predicates

```
/bib/book/author[first-name]
```
Result: <author> <first-name> Rick </first-name>
<last-name> Hull </last-name> </author>

XPath: More Predicates

```
/bib/book/author[first-name]/address[ZIP][city]/last-name
```
Result: <last-name> ... </last-name>
<last-name> ... </last-name>

How do we read this?
First remove all qualifiers (predicates):
`/bib/book/author[last-name]`

Then add them one by one:
`/bib/book/author[first-name][address][last-name]` etc

XPath: More Predicates

```
/bib/book[@price < 60]
```
```
/bib/book[author/@age < 25]
```
```
/bib/book[author/text()]
```
XPath: Position Predicates

  The 2nd book

- /bib/book[last()]  
  The last book

- /bib/book[@year = 1998][2]  
  The 2nd of all books in 1998

  2nd book IF it is in 1998

XPath: More Axes

- . means current node  
  /bib/book[.//review]

- Same as  
  /bib/book[review]

- /bib/author/.//first-name  
  Same as  
  /bib/author/first-name

XPath: More Axes

- .. means parent node  
  /bib/author/..//author/zip  
  Same as  
  /bib/author/zip

- /bib/book[.//review/..//comments]  
  Same as  
  /bib/book//comments[review]

- Hint: don't use ...

XPath: Summary

- bib  
  matches a bib element

- *  
  matches any element

- /  
  matches the root element

- bib  
  matches a bib element under root

- bib/paper  
  matches a paper in bib

- bib/paper  
  matches a paper in bib, at any depth

- /paper  
  matches a paper at any depth

- paper/book  
  matches a paper or a book

- @price  
  matches a price attribute

- bib/book/@price  
  matches price attribute in book, in bib

- bib/book[@price="55"]  
  matches price attribute in book, in bib

- bib/book[@price="55"]  
  matches...

- bib/book[@price="55" or @price="99"]  
  matches...

CSE 444 - Autumn 2010